Dynamic computing random access memory

Dynamic Computing Random Access Memory

F. L. Traversa[1,2], F. Bonani[3], Y. V. Pershin[4], M. Di Ventra[2]

[1] Department of Electronic Engineering, Universitat Autònoma de Barcelona, Spain
[2] Department of Physics, University of California, San Diego, La Jolla, California 92093-0319, USA
[3] Dipartimento di Elettronica e Telecomunicazioni, Politecnico di Torino, 10129 Torino, Italy
[4] Department of Physics and Astronomy, University of South Carolina, Columbia, South Carolina 29208, USA

E-mail: fabio.traversa@polito.it, fabrizio.bonani@polito.it, pershin@physics.sc.edu, diventra@physics.ucsd.edu

Abstract. The present von Neumann computing paradigm involves a significant amount of information transfer between a central processing unit (CPU) and memory, with concomitant limitations in the actual execution speed. However, it has been recently argued that a different form of computation, dubbed *memcomputing* [*Nature Physics* **9**, 200-202 (2013)] and inspired by the operation of our brain, can resolve the intrinsic limitations of present day architectures by allowing for computing *and* storing of information on the *same* physical platform. Here we show a simple and practical realization of memcomputing that utilizes easy-to-build memcapacitive systems. We name this architecture Dynamic Computing Random Access Memory (DCRAM). We show that DCRAM provides *massively-parallel* and *polymorphic* digital logic, namely it allows for different logic operations with the same architecture, by varying only the control signals. In addition, by taking into account realistic parameters, its energy expenditures can be as low as a few fJ per operation. DCRAM is fully compatible with CMOS technology, can be realized with current fabrication facilities, and therefore can really serve as an alternative to the present computing technology.

PACS numbers:

1. Introduction

There is currently a surge of interest in alternative computing paradigms [1] that can outperform or outright replace the present von Neumann one [2]. It is clear that such alternatives have to fundamentally depart from the existing one in both their execution speed as well in the way they handle information. For at least a couple of decades, quantum computing [3, 4] (QC) has been considered a promising such alternative, in view of its intrinsic massive parallelism afforded by the superposition principle of quantum mechanics. However, the range of QC applications is limited to a few problems such as integer factorization [5] and search [6].

In order to obtain a paradigm shift we then need to look somewhere else but no farther than our own brain. This amazing computing machine is particularly suited for massively-parallel computation. It is polymorphic, in the sense that it can perform different operations depending on the input from the environment, and its storing and computing units – the neurons and their connections, synapses [7] – are the *same* physical object. Such a brain-inspired computing paradigm has been named *memcomputing* [8] and relies on resistors [9, 10], capacitors or inductors with memory (collectively called memelements) [11, 13] both to store the data and to perform the computation. The features of memelements that make them very attractive from a practical point of view are: *i*) they are a natural by-product of the continued miniaturization of electronic devices, and *ii*) they can be readily fabricated [12, 14, 15, 16] making memcomputing a realistic possibility.

This work reports a memcomputing implementation based on solid-state memcapacitive systems [17] (capacitors with memory). While previous memcomputing schemes [18, 19, 20, 21, 22, 23, 24, 25] employ intrinsically dissipative memristive devices [9, 10] (resistors with memory), we take advantage of very low power dissipation in memcapacitive systems [11] to build a *Dynamic Computing Random Access Memory* (DCRAM) capable of storing information and performing polymorphic logic computation. This new platform allows for massively-parallel logic operations directly in memory thus offering a starting point for a practical solution to the von Neumann bottleneck problem [26]. Moreover, we would like to emphasize that our idea is not limited to the specific type of memcapacitive systems used for model calculations reported in this work. For example, ferroelectric capacitors [27] used in FERAM [28] and currently evaluated for new DRAM solutions are also promising candidates for DCRAM.

While the general topology of DCRAM (Fig. 1) is similar to that of conventional dynamic random access memory (DRAM), its memory cells are solid-state memcapacitive systems [17]. These are multilayer structures composed of insulating layers (three in the particular realization we consider here) alternated by metal layers. The most external insulating layers are made of high-κ materials with very high potential barrier so that negligible charge can pass trough them. On the other hand, the intermediate layer is formed out of a low-κ material with low potential barrier. This

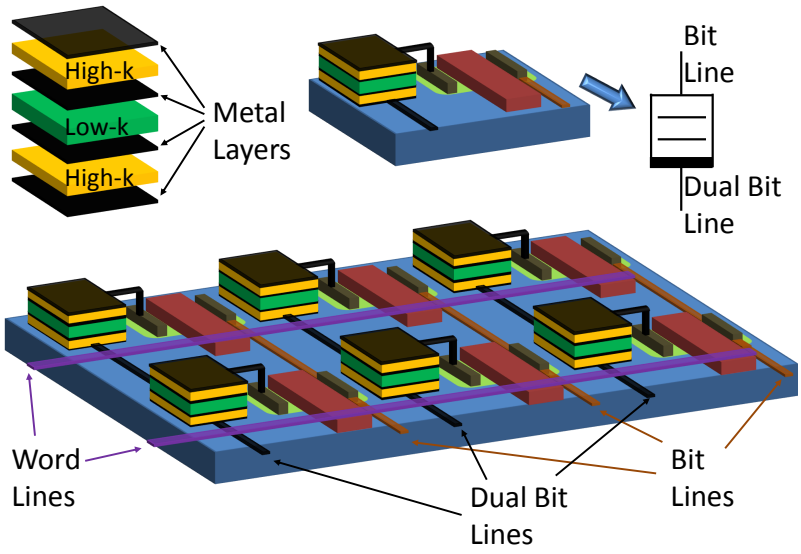


Figure 1. Possible realization of DCRAM. The memory cell (in the top right corner) is a solid-state memcapacitive system composed by three insulating layers separated by metal layers. Two-dimensional DCRAM circuit (bottom) is composed by an array of cells having an access element (MOSFET) with a gate controlled by the word line. In order to perform READ or WRITE operations with a given cell, a positive voltage is applied to its word line, ground to its dual bit line, and suitable voltage pulses to its bit line. For computation purposes, several cells can be coupled through bit and dual bit lines as described in the text.

choice allows for non-negligible charge migration between two internal metal layers at appropriate bias conditions. If the middle insulator layer is thin enough, the internal charge current is due to quantum tunnelling [29] and can be easily tuned over a wide range of values [30].

Although no prototype of solid-state memcapacitive systems has been realized yet, we point out that its actual realization oriented to VLSI circuits may not be of a simple planar geometry. In fact, DRAM capacitors are normally of cylindrical shape. Consequently, a possible realization of solid-state memcapacitive systems could consist of three cylindrical capacitors forming an effective solid-state memcapacitive system.

2. Example of memcapacitor structure and device optimization

The capacitance C_d of the solid-state memcapacitive system we consider here is defined using the standard relation $q = C_d V_C$, where q is the charge on the capacitor plates (external metal layers) and V_C is the voltage applied to the system. Importantly, C_d is a function of the internal state, namely, it depends on the ratio Q/q where Q is the internal charge (see the top left inset in Fig. 4 below) [17]. Moreover, C_d can diverge and take negative values [17] leading to a variety of transient responses.

The internal memory of the memcapacitive system [17] arises from the delay of the internal charge response caused by a tunneling barrier of the central insulator layer [17].

The tunneling barrier can be lowered by a voltage bias applied to the capacitor plates. In this case, a finite internal current (between the internal metal layers) changing Q is possible. The internal charge Q becomes trapped when the shape of the potential barrier is restored. Therefore, the applied voltage pulses can be used to control the internal charge Q, which can be subsequently stored.

Here we discuss the features of the solid-state memcapacitor as proposed in [17], using realistic values of parameters compatible with the 2012 International Technology Roadmap for Semiconductors (ITRS) specifications [1]. From ITRS 2012, the capacity of DRAM cell is about $20 - 25$ fF and the equivalent oxide thickness (EOT) is 0.5 nm for a high-k material of $k = 50$. A rapid calculation shows that the area of the metallic layers of an *equivalent* planar capacitor (common geometries for DRAM capacitors are not in general planar, several complex geometries, e.g., cylindrical or pedestal structures, are employed by different manufacturers) has to be of the order of 0.25 μm^2, so we use this value in our simulations. Moreover, the physical thickness of the insulator, from the EOT and $k = 50$, ranges between $6 - 10$ nm. Using these data, we consider the memcapacitor structure sketched in Fig. 1. The thickness of the two high-k insulators is supposed to be 6 nm and we assume they are made of standard modern high-k material (e.g. TiO_2) with $k = 50$. Finally in our simulations we consider transmission lines with common values for DRAM fabrication, i.e., $R = 1.5$ kΩ mm^{-1} and $C = 0.2$ pF mm^{-1} for a length of 1 mm.

Physical parameters (thickness and k value) of the low-k layer require a more careful consideration since the lifetime of Q strongly depends on these two parameters. Let us then focus on the *storage mode*, namely, the situation that follows a WRITE operation (application of 1 ns voltage pulse of certain amplitude). In order to model the least favorable conditions such as a strong external leakage current (due to imperfect switches and other processes), we assume $V_C = 0$ irrespective of the written bit. This choice is different from that in common DRAM where, in the storage mode, $V_C > 0$ if the stored bit is 1 and $V_C = 0$ if it is 0. Our main goal here is to evaluate the possibility of information storage on long time scales compared to typical DRAM decay times using, however, DRAM-like chip structure.

Let us consider a physical model of solid-state memcapacitive system with a barrier height of 0.2 eV for the low-k material and infinite barrier for the high-k one. The equations governing the time variation of Q and q can be written as [17]

$$V_C = \frac{Q}{C_2} + \frac{q}{C_0} \tag{1}$$

$$\frac{dQ}{dt} = -I(Q + q) \tag{2}$$

where I is the tunnel current through the low-k material, C_0 is the (constant) capacitance of the total memcapacitive system (with respect to q only) and C_2 is the capacitance of the internal capacitor composed by the low-k material and internal metal layers. If the barrier is sufficiently thin then the current can be approximated by the Simmons formula [30].

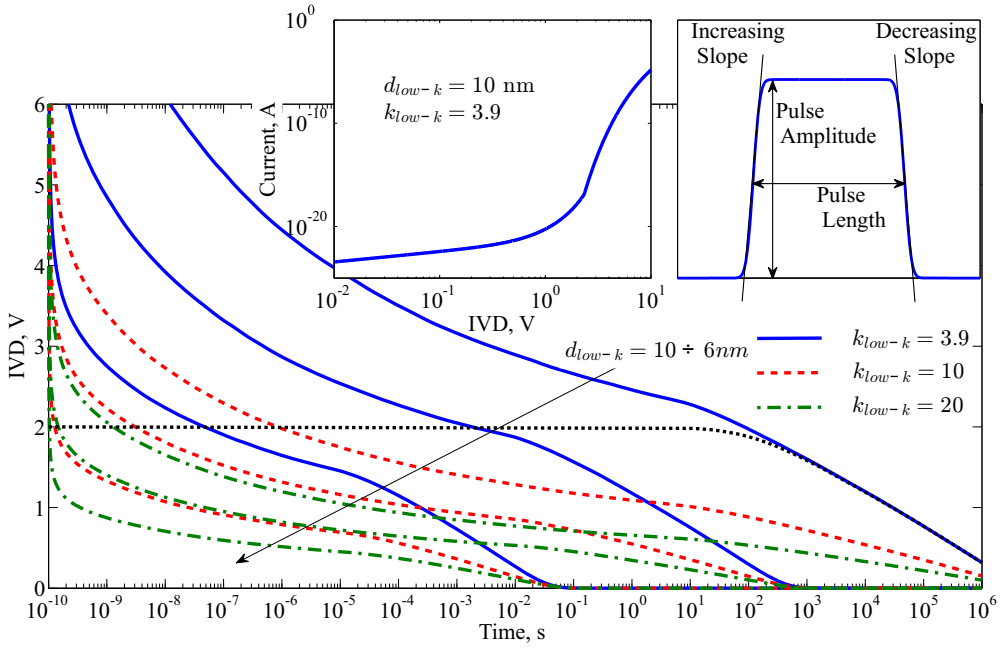


Figure 2. Decay of the internal voltage difference (IVD) Q/C_2 for different thicknesses and dielectric constants of the middle insulator. This plot shows envelope curves of the internal charge decay that can be used to track the long-time behavior for any initial value of Q. As an example, the dotted black curve represents the decay of IVD Q/C_2 for the initial condition $Q/C_2 = 2$V at $k_{low-k} = 3.9$ and $d_{low-k} = 10$ nm. Note, that this curve converges with the corresponding envelope at longer times. The top left inset reports the current $I((1 - C_0/C_2)Q)$ versus Q/C_2. Top right inset presents the shape of the voltage pulse with 10 V/ns rising and falling edges.

Taking into account that $C_0 < C_2$ and I is monotonous with a unique 0 at $Q+q = 0$, there is unique steady-state solution $Q = q = 0$ at $V_C = 0$. The top inset in Fig. 2 shows that the current $I(Q + q)$ is very small at smaller values of $Q + q$ suggesting the possibility of quite low charge relaxation rate at nonzero Q. At $V_C = 0$, Eq. (2) can be rewritten as

$$\frac{dQ}{dt} = -I((1 - C_0/C_2)Q). \tag{3}$$

This equation describes the decay of the internal charge Q in the storage mode. Fig. 2 shows the decay of Q for several values of k and layer thicknesses. It is worth noticing that at certain values of parameters, such as the thickness of 10 nm and $k = 3.9$, the information is stored for a long time. In fact, after 10^6 s (about 11.5 days) a reasonable amount of charge still remains in the memcapacitive system. Thus, modifying the parameters of the memory cell (the layer thickness, dielectric constant or even the barrier height) one can select an appropriate lifetime of the internal charge Q.

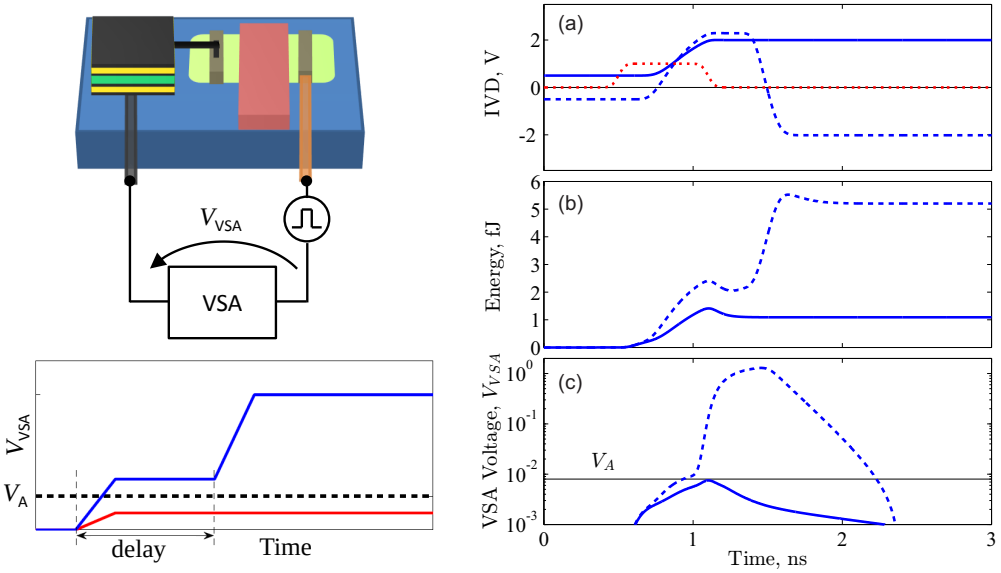


Figure 3. Configuration and simulation of READ-REFRESH process. The circuit configuration for this process is presented on the left. It consists of a memory cell connected to a pulse generator and VSA. The bottom left plot shows the ideal VSA response when its input signal is below (red line) and above (blue line) its threshold. The simulation of a READ-REFRESH process for the initial condition of a partially decayed bit ($Q_i/C_2 = \pm 0.5$ V) are given on the right. Here, the circuit is driven by 0.5 ns length, 1 V amplitude voltage pulse during the delay time of VSA. We report (a) the time variation of the normalized charges Q/C_2 (solid and dashed blue lines) and the voltage pulse (dotted red line), (b) the dissipated energy, and (c) VSA output.

3. WRITE and READ operations.

In our scheme, the binary information is encoded in the internal charge Q of the memcapacitive system. It is assumed that $Q \geq Q_r$ corresponds to logic 1, $Q \leq -Q_r$ corresponds to logic 0, and the logic value is not defined when $-Q_r < Q < Q_r$. The threshold Q_r is introduced to reliably distinguish logic values, and as such is defined according to the sensitivity of the voltage sense amplifiers (VSA) that we exploit to allow for the bit value detection.

When a voltage pulse is applied to a memory cell, its current response strongly depends on its internal charge Q. We thus use this current response to read the information stored in the memory cell: The common solution (widely used in consumer electronics including standard DRAM technology) employs VSAs.

As depicted in Fig. 3, the VSA is connected to the memory cell in series with a voltage pulse generator. The ideal characteristics of the VSA are presented in the left bottom inset of Fig. 3. It is important to know that VSA amplifies the response voltage V_{VSA} if $V_{VSA} > V_A$, where V_A is a certain threshold voltage. Generally, the delayed response of VSAs is associated to the internal capacitances of the Metal-Oxide-Semiconductor (MOS) structures they are made of. During the delay time, the voltage

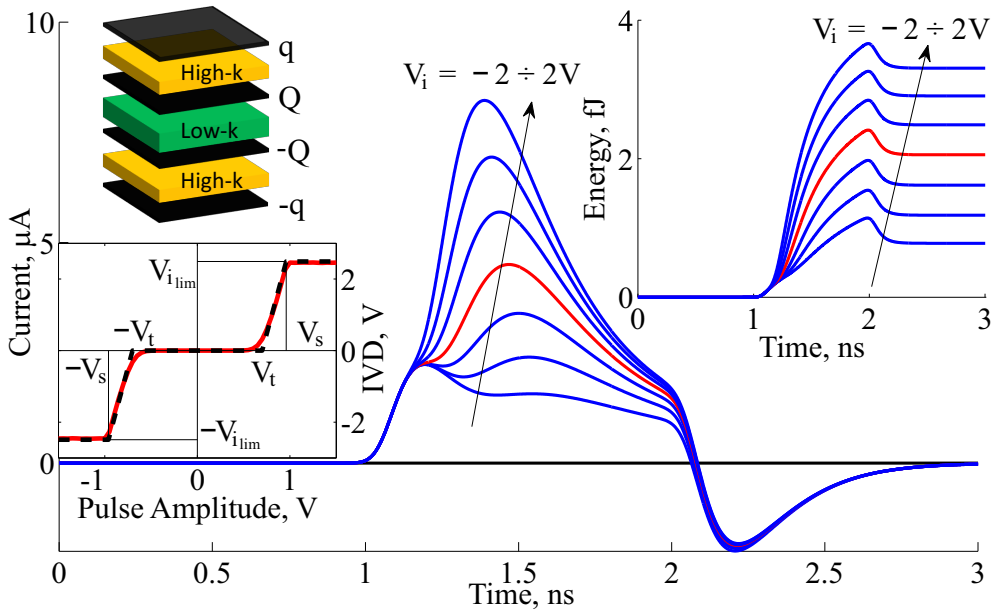


Figure 4. Single cell response to a voltage pulse under READ/WRITE conditions as described in Fig. 1. In our simulations, the bit and dual bit lines are modeled as transmission lines with typical parameters for DRAM $R = 1.5$ kΩmm^{-1} and $C = 0.2$ pFmm^{-1} assuming 1 mm line length. The voltage pulse is a smooth square pulse of 1 V amplitude and 1 ns width starting at $t = 1$ ns. The main graph is the current response measured at the end of the bit line for several initial values of the internal charge Q. The red line refers to $Q = 0$ initial condition. To quantify Q, an effective internal voltage difference (IVD) is defined as $V_i = Q/C_2$ with C_2 the geometrical capacitance of the intermediate layer, $C_2 = A\varepsilon_0 k_{low-k}/d_{low-k}$, where A is the surface area, ε_0 is the vacuum permittivity, k_{low-k} is the relative permittivity of the central layer, and d_{low-k} is its thickness. The top right inset shows the cell's dissipated energy. Bottom left inset: the effective internal voltage difference as a function of voltage pulse amplitude in 1 s after the voltage pulse application.

pulse generator induces the response voltage V_{VSA}. Being amplified, V_{VSA} provides the value stored in the memory cell.

The WRITE, READ and logic operations with memcapacitive memory cells are performed with the help of control circuitry that provides appropriate signals. In order to make the discussion even more realistic, the parameters we have used throughout the simulations belong to the ITRS 2012 standards [1]. Simulations have been carried out using the general purpose in-house NOSTOS (NOnlinear circuit and SysTem Orbit Stability) simulator developed by one of the authors (FLT) initially for studying circuit stability [31, 32], and recently extended to analyze circuits including memory elements [33]. Let us consider the WRITE operation first. For this purpose, we employ the circuit configuration shown in the top right corner of Fig. 1 where the dual bit line (DBL) is grounded and the voltage pulse is applied to the bit line (BL). As it is mentioned above, applied voltage pulse lowers the potential barrier between the internal metal layers allowing for an internal charge redistribution.

An important observation that one can make at this point is that the WRITE process is of the threshold type. Indeed, one can define a threshold voltage V_t such that there is no significant charge transfer between the internal plates at applied voltage amplitudes below V_t (see the bottom left inset in Fig. 4). On the contrary, at pulse amplitudes exceeding V_t a considerable amount of charge can tunnel between the internal layers. In our device structure, V_t is about 0.5 V, which is much larger than the perturbations usually induced by MOS transistor leakage currents. Moreover, the existence of V_t results also in an internal charge saturation shown in the bottom left inset of Fig. 4.

Next, let us consider the READ operation in DCRAM. Similarly to DRAM, the READ process is destructive (see the top right plot of Fig. 3: when the voltage pulse acts, the information inside the memory cell is destroyed since the final state inside the memory cell is 1) and thus needs to be followed by a REFRESH operation. In order to have a better understanding, we consider the current response shown in Fig. 4. One can notice significant variations in the cell response depending on the initial value of Q. These differences are used to measure the logic value stored in the cell with VSAs similarly to DRAM technology. However, VSA amplifies a voltage difference above or below a certain voltage threshold. To meet the VSA modus operandi, the current response can be transformed into the voltage response connecting the bit and dual bit lines to VSA input terminals. As the voltage pulse used in READ changes the internal charge Q, a suitable REFRESH operation is applied after the READ.

In summary, the sequence consists of two steps. First, a voltage pulse (in our simulations, of 0.5 ns length and 1 V amplitude) is applied by the generator. It produces a voltage response that is considered as input for VSA during its "delay state". Subsequently, if $V_{VSA} > V_A$ the VSA amplifies the voltage V_{VSA} and 0 is written, on the contrary, if $V_{VSA} < V_A$ the VSA does not act and 1 is written. Fig. 3 reports simulations of the READ-REFRESH process considering an extreme case of a partially decayed bit showing all the features mentioned above. Moreover, we would like to emphasize that the dissipated energy has a significant dependence on the value of bit (0 or 1). This is due to an asymmetry in VSA response. In fact, when $V_{VSA} > V_A$ (VSA is activated) the dissipated energy is about 5 fJ. In the opposite case (initial value is 1) this energy is about 1 fJ.

The top right inset of Fig. 4 shows the dissipated energy when a pulse of 1 ns length and 1 V amplitude is applied. In fact, this calculation gives a reference for the order of magnitude of the dissipated energy for all DCRAM operations (WRITE, READ, COMPUTATION) because of close operating conditions. It is worth noticing that this energy is of the order of few fJ, comparable to the best cases of extremely low-energy storage and computation [34], and computation only with CMOS architectures [34]. Importantly, the information is stored directly in DCRAM saving the power usually needed to transfer it to/from the CPU.

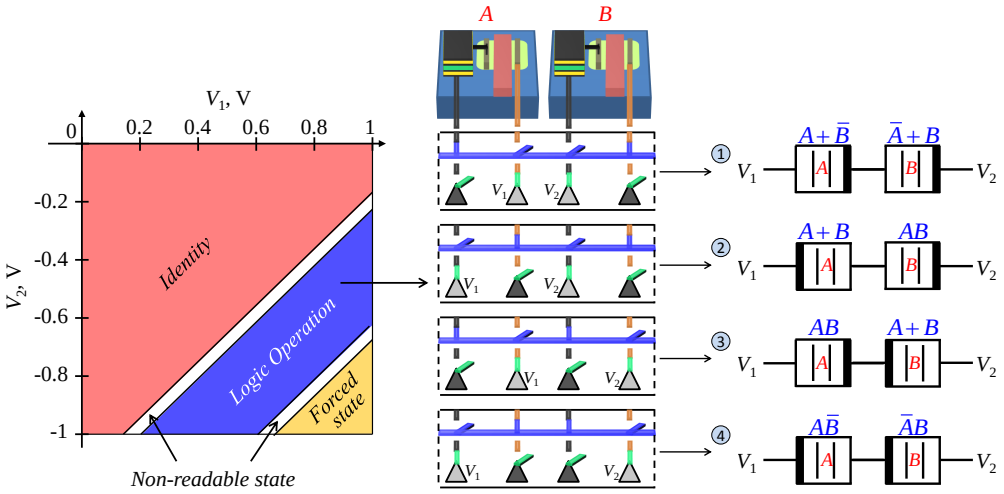


Figure 5. Map of logic operations. Two memory cells can be connected in four different ways giving rise to four logic operations. The symbols + and ¯ are the OR and NOT operation respectively, while the AND operation is the implicit multiplication. Here, V_1 and V_2 are amplitudes of voltage pulses applied to the external connections of the coupled memory cells. Depending on these amplitudes, there are several regions in the logic map. Amplitudes belonging to the *identity* region do not change initial values in memory cells. Amplitudes belonging to the *logic operation* region perform computation as in the scheme to the right. Amplitudes belonging to the *forced state* region change the initial values to 1 or 0 depending on device coupling order and polarity. Amplitudes belonging to the *non-readable state* region produce an intermediate (non readable) internal states with $-Q_r \leq Q \leq Q_r$.

4. Polymorphic computation.

Let us consider the simplest realization of logic gates when 2 memory cells are used to store the input and (after the computation) the output values. For computation purposes, these memory cells are coupled as shown in Fig. 5 using appropriate switches at the end of the BL and DBL. As shown in Fig. 5, the dynamics of the internal charges Q of two coupled cells subjected to a couple of synchronized voltage pulses depends on the initial combination of internal charges of these cells. In this way, the final values of the internal charges can be considered as a result of a logic operation over initial values stored at $t = 0$ in these cells (see fig. 5). As a specific example, let us consider configuration 2 from Fig. 5 assuming that -0.73 V and 0.73 V amplitude voltage pulses are applied to the memory cells. Fig. 6 demonstrates the evolution of Q for both cells. Notice that the final values of Q in cells A and B realize OR and AND gates, respectively. The dissipated energy (bottom plots in Fig. 6) is quite low: it is less than 2 fJ in the worst case scenario, and, in the case of (1, 0) initial configuration, it is much lower. However, it is worth noticing that, after computation (see Fig. 6), the bits stored in the cells are only partially written: the computation process must be completed by a REFRESH process, thus increasing the total required energy per operation by a few fJ, depending on the actual realization of VSA.

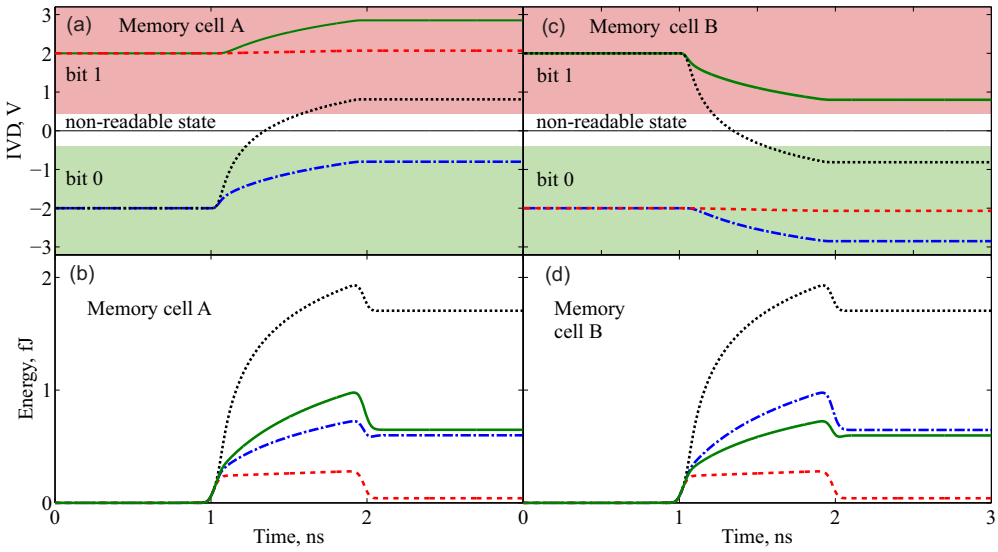


Figure 6. Time variation of IVD and dissipated energy for the second logic gate of figure 5. The voltage pulse amplitudes are $V_1 = 0.73V$ V and $V_2 = -0.73$ V, and the pulse length is 1 ns. The evolution of IVD for both memory cells at different initial conditions are shown by different line styles in (a) and (c). The dissipated energy is plotted in (b) and (d).

Considering possible connections and device polarities one can find that two coupled cells can be used to form a (redundant) basis for a *complete set of logic operations*. In fact, it is known [35, 22] that combining only AND and NOT or OR and NOT functions, any logic expression can be evaluated. In our case, with two coupled memory cells we can in fact perform 6 different two-bit operations, depending both on how the cells are coupled, and on the amplitudes of the applied voltage pulses. Therefore, these two coupled memory cells form universal logic gates as it is exhaustively proved below.

The universal gate offered by the DCRAM architecture is not its only advantage. DCRAM is capable of intrinsically *parallel* computation. In fact, after only one computation step, we find a different output on each memcapacitive system: this means two operations at the same time. As shown later, by connecting three memory cells and varying the pulse amplitudes and the connection topology, we can perform even more complex operations in one step, and obtain three different outputs written into each memory cell. More importantly, one can perform simultaneous operations over multiple groups of two or three coupled cells. We also point out that by using only one of the possible connection topologies of three memory cells, we obtain another universal gate for two-bit computation with *fixed* connection topology representing a possible solution to avoid the supplemental circuitry needed for dynamic connections.

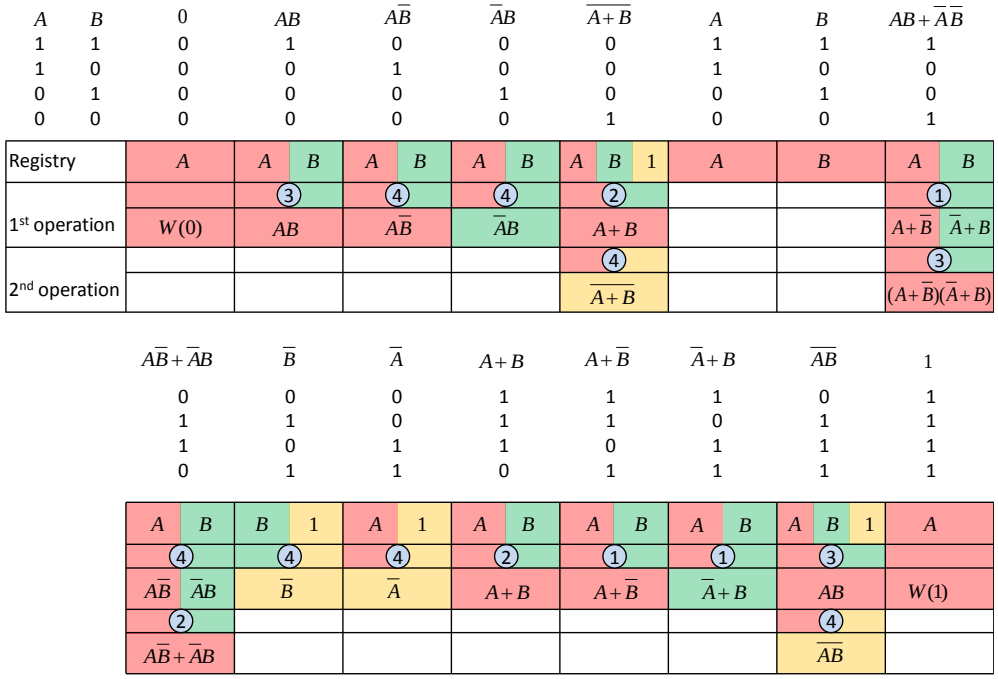


Figure 7. Two-bit logic functions. The additional bit set to 1 is used for negation. Circled numbers refer to logic operation of figure 5. Colors denote the memory cell involved in operation and the cell storing the output. W(1) and W(0) stand for the operation WRITE 1 and 0, respectively.

4.1. Two-bit and Three-bit operations with dynamic connections

Parallel logic operations performed by DCRAMs, summarized in figure 5, can be used to define logic gates forming a (redundant) complete basis for any boolean logic function. In order to prove this claim, figure 7 shows how to perform all possible two-bit logic operations using DCRAM gates. We notice that in the worst case scenario, a three-bit registry (three cells) is needed (the third bit, initially set to 1, is used to perform negation), and a two-level operation is required. Compared with CMOS NAND logic or NOR logic, DCRAM logic circuits require less components. In fact the commonly used CMOS NAND or NOR logic gates require up to 5-level operation scheme, and up to 20 transistors to perform the same set of two-bit functions.

Using the same scheme, we can perform any n–bit operation exploiting 2–bit gates only. Here, we make some considerations on three-bit operations, for which a complete treatment is possible. Using a 5–bit registry made of the A, B and C inputs and two additional bits, one set to 1 employed for negation and the other equal to one of the three inputs A, B or C (depending on the desired logic function), any three-bit logical operation can be performed using at most a 4–level operation scheme (Fig. 8). In figure 9–(b) an example of 4−level three-bit operation is shown. In this case, the registry is composed by the three inputs (A, B and C) and only one additional bit (in this case A), because no bit for negation is required. In figure 9–(a), the comparison with a

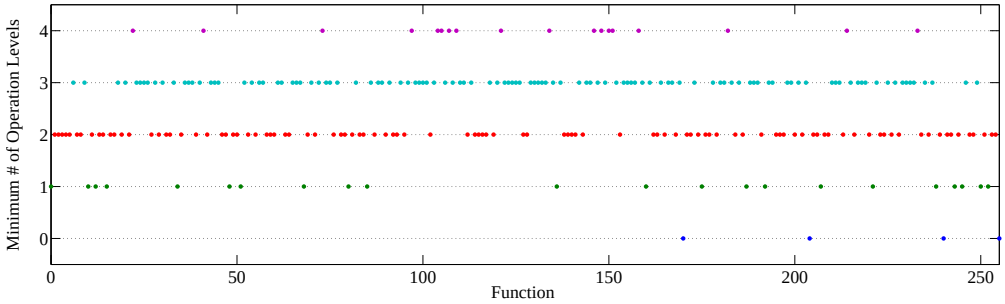
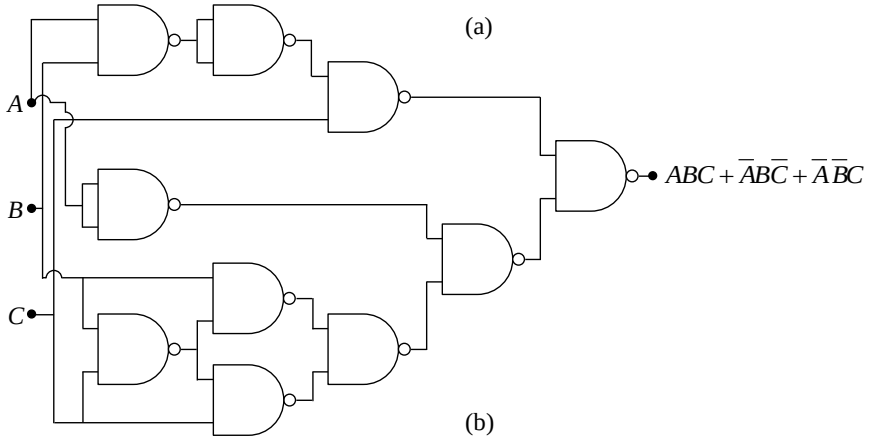


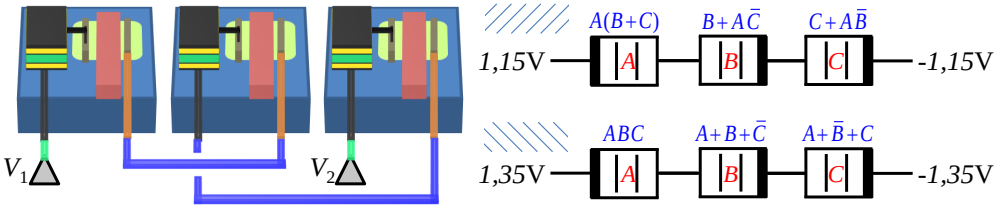
Figure 8. Number of operation levels for any three-bit boolean function. There are 2^8 possible boolean functions involving three bits, so in the $x-$axis each function is coded using the equivalent decimal number.



Registry	A	B	C	A
1st operation	③	③	③	③
	AB	$A+B$	AC	$A+C$
2nd operation	③	④	③	④
	$ABAC=ABC$	$(A+B)(\overline{A+C})=\bar{A}B\bar{C}$	$AB+AC=A(B+C)$	$(\overline{A+B})(A+C)=\bar{A}\bar{B}C$
3rd operation	②	②		
	$ABC+\bar{A}B\bar{C}$	$\bar{A}B\bar{C}ABC=0$	$A(B+C)$	$\bar{A}\bar{B}C$
4th operation	②			②
	$ABC+\bar{A}B\bar{C}+\bar{A}\bar{B}C$	0	$A(B+C)$	$(ABC+\bar{A}B\bar{C})\bar{A}\bar{B}C=0$

Figure 9. (a) CMOS-NAND logic circuit for the three-bit operation $ABC + \bar{A}B\bar{C} + \bar{A}\bar{B}C$. (b) DCRAM 4–level scheme for the same logic function.

two-input NAND logic (possibly using programmable digital circuits) is reported. It is worth noticing that, using CMOS NAND logic, the same operation is performed within a 5−level operation scheme using 10 NAND gates, i.e., 40 transistors, thus proving that the complexity of the CMOS circuit is much higher than for our DCRAM implementation.



Registry		1	A	B
1st operation level	operation	$A+B$	$A+\bar{B}$	$\bar{A}+B$
	W/R	$W(1)$	REFRESH	REFRESH
2st operation level	operation	$(A+\bar{B})(\bar{A}+B)$	1	1
	W/R	REFRESH	Nothing	Nothing

Figure 10. Computation of the logic function $AB+\bar{A}\bar{B}$ using three connected memory cells. The topology of the connections is represented in the top left of the figure. The two gates obtained varying the pulse amplitude are sketched on the top right of the figure and indicated by the two different textures on the left of the gates.

A	B	A	B	0	AB	$A+B$	$A\bar{B}$	$\overline{AB}$	$A\bar{B}+\bar{A}B$	$AB+\bar{A}\bar{B}$
1	1	1	1	0	1	1	0	0	0	1
1	0	1	0	0	0	1	1	1	1	0
0	1	0	1	0	0	1	0	1	1	0
0	0	0	0	0	0	0	0	1	0	1

Registry	A	B	0	A	B	0	A	B	0	A	B	1
1st operation level				AB	$A+B$	$A\bar{B}$	AB	$A+B$	$A\bar{B}$	AB	$A+B$	1
				R	R	R	R	W(0)	W(0)	R	R	W(0)
2nd operation level							AB	AB	AB	0	1	$AB+\bar{A}\bar{B}$
							W(1)	W(0)	R	W(1)	W(0)	R
3rd operation level							AB	$\overline{AB}$	1	$AB+\bar{A}\bar{B}$	$A\bar{B}+\bar{A}B$	1
							R	R	N	R	R	N

$\bar{A}$	$A+\bar{B}$	$\bar{A}+B$	$\overline{AB}$	$\overline{A+B}$
0	1	1	0	0
0	1	0	0	0
1	0	1	1	0
1	1	1	0	1

1	A	0	0	A	B	0	A	B	A	B	0
A	1	$\bar{A}$	0	$A+\bar{B}$	$\bar{A}+B$	0	$A+\bar{B}$	$\bar{A}+B$	AB	$A+B$	$A\bar{B}$
N	R	R	N	R	R	W(1)	R	W(0)	W(1)	R	W(0)
						$A+\bar{B}$	1	$\bar{A}B$	$A+B$	1	$\bar{A}+B$
						R	N	R	R	N	R

Figure 11. Two-bit logic functions. The configuration of the connections for the three-memory cell polymorphic gate is the same as that in fig. 10. The textures indicate the gate kind as in fig. 10 (depending on the pulse amplitudes). W(1) and W(0) stand for the operation WRITE 1 and 0, respectively, and R = REFRESH. The functions $\bar{B}$ and 1 are not reported for sake of conciseness because they can be simply obtained as in the fifth column for $\bar{A}$ and in the first column for 0, respectively.

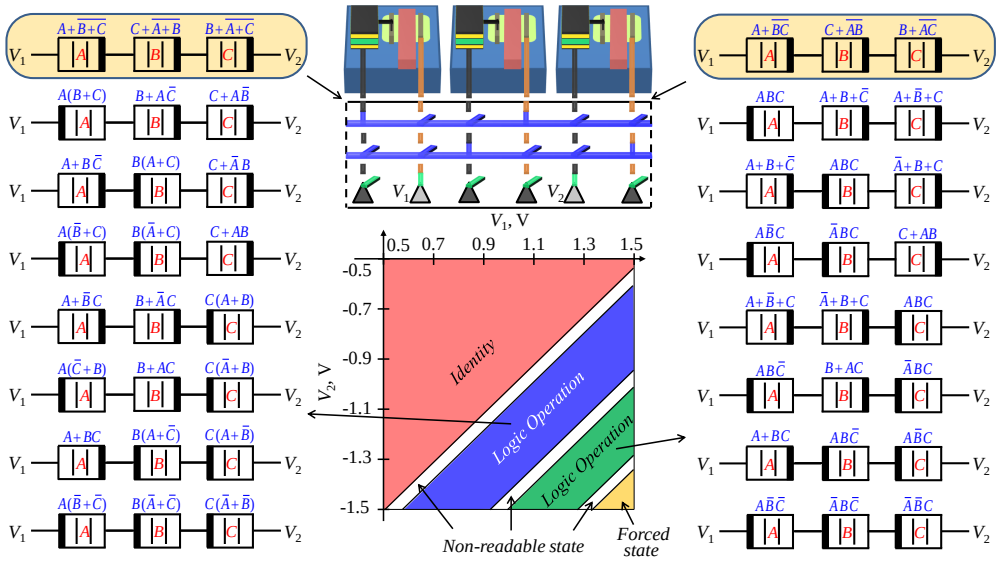


Figure 12. Logic gates with three coupled cells. In the center, we show a map of operations as a function of amplitudes of pulses applied to the external connections of the coupled memory cells. Depending on these amplitudes, there are several regions in the logic map. Amplitudes belonging to the *identity* region do not change initial values. Amplitudes belonging to the *logic operation* region realize logic functions presented in schemes to the right and left. Amplitudes belonging to the *forced state* region change the initial values to 1 or 0 depending on device coupling order and polarity. Amplitudes belonging to the *non-readable state* region produce an intermediate (non readable) internal states with $-Q_r \leq Q \leq Q_r$. The symbols $+$ and $^-$ are the OR and NOT operations, respectively, the implicit multiplication is the AND.

4.2. Fixed connection two-bit operations

Finally, we consider the three-bit gate presented in fig. 10. We assume a configuration with *fixed* connections (while computation is performed). As shown in figure fig. 10, varying the pulse amplitudes applied to the cells we can obtain two different logic outputs for each memory cell. We define these as the logic outputs of the first and second kind. Moreover, at each computation step the REFRESH and WRITE processes are performed to prepare the cells for the next computation step. The bits 1, A and B are initially written in the three memory cells (registry). Then, we apply the synchronized voltage pulses V_1 and V_2 with amplitude 1.15 V and -1.15 V, respectively, to obtain the gate of the first kind. The first-level operation is completed by the REFRESH of the second and third memory cells and by writing 1 in the first one. Then, the second-level operation implements the gate of the second kind, and the boolean function $AB + \bar{A}\bar{B}$ is obtained.

Using the processes described above, we can set up a universal gate capable to perform any two-bit logic operation without changing the topology of the circuit. For example Fig. 11 shows how to obtain all possible 2-bit logic functions using the 3-bit fixed polymorphic gate of fig. 10. Finally, fig. 12 reports the variety of 3-bit polymorphic

logic gates that can be implemented using three coupled memory cells. In this case it is evident the separation into two regions of applied voltage amplitudes providing polymorphism without changing the connection topology.

5. Conclusions

In conclusion we have introduced a simple, practical, and easy-to-build memcomputing architecture that processes and stores information on the same physical platform using two-terminal passive devices (memcapacitive systems). Being low-power, polymorphic and intrinsically massively-parallel, DCRAM can significantly improve computing capabilities of the present day von Neumann architecture. This is performed by transferring a significant amount of data processing directly into the memory, where the data is stored. Although it is still an open question which specific algorithms will mostly benefit from such an approach, we expect that our scheme will be extremely useful in scientific calculations, image and video processing and similar tasks.

In order to make a specific estimation of computation speed-up using our approach, let us compare a performance of a traditional personal computer equipped with typical DRAM chips with this of a DCRAM-based computer. For example, consider a 4 GB memory system, with two 2 GB ranks, each consisting of eight 256 MBx8, 4-bank devices [36]. Moreover, each of the 4 banks in a 256 MB device is split into 8 arrays of 8 MB each. If there are 65,536 rows of 1024 columns of bits in each array, a row access provides a selection of 1024 bits per array, giving a total of 65,536 bits across 8 chips of 8 arrays each. This is the number of bits that can be involved simultaneously in a *single* parallel calculation using DCRAM, which lasts for about 20 ns (accounting for a 4-level computation) as discussed above (here we assume that all 65,536 bits are grouped into small few-bits circuits at each calculation step). On the other hand, a standard CPU processes 64 bits per each clock cycle. Accounting for the memory access time of 10 ns [37], we can conclude from this simple example that a DCRAM could be in principle up to 1000 times faster than the usual Von Neumann architecture.

Finally, we emphasize again that an actual realization of DCRAM is not limited to the employment of solid-state memcapacitive systems considered in this work. Other memcapacitive systems could serve as even better solutions for practical implementations of DCRAM. We thus hope that our results will be of interest to a wide community of researchers and lead to the next generation of brain-like computing memory.

6. Acknowledgment

This work has been partially supported by the Spanish Project TEC2011-14253-E, NSF grants No. DMR-0802830 and ECCS-1202383 and the Center for Magnetic Recording Research at UCSD.

References

[1] http://www.itrs.net/

[2] Von Neumann, J. First draft of a report on the EDVAC. *IEEE Annals Hist. Comput.* **15**, 27-75 (1993).

[3] Ladd, T. D., Jelezko, F., Laflamme, R., Nakamura, Y., Monroe, C. & O'Brien, J. L. Quantum computers. *Nature* **464**, 45-53 (2010).

[4] Nielsen, M. A., Chuang, I. L. *Quantum computation and quantum information* (Cambridge University Press, 2000).

[5] Shore, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.* **26**, 1484-1509 (1997).

[6] Grover, L. K. Quantum Mechanics Helps in Searching for a Needle in a Haystack. *Phys. Rev. Lett.* **79**, 325-328 (1997).

[7] Cowan, M. R., Maxwell, W., Sudhof, T. C. & Stevens, C. F. *Synapses* (Johns Hopkins University Press, 2003).

[8] Di Ventra, M. & Pershin, Y. V. The parallel approach. *Nature Physics* **9**, 200-202 (2013).

[9] Chua, L. O. Memristor-The missing circuit element. *IEEE Trans. Circuit Theory* **18**, 507-519 (1971).

[10] Chua, L. O. & Kang, S. M. Memristive devices and systems. *Proc. IEEE* **64**, 209-223 (1976).

[11] Di Ventra, M., Pershin, Y. V. & Chua, L. O. Circuit elements with memory: memristors, memcapacitive systems and meminductors. *Proceedings of the IEEE* **97**, 1717-1724 (2009).

[12] Pershin, Y. V. & Di Ventra, M. Memory effects in complex materials and nanoscale systems. *Advances in Physics* **60**, 145-227 (2011).

[13] Di Ventra, M. & Pershin, Y. V. On the physical properties of memristive, memcapacitive, and meminductive systems. *Nanotechnology* **24**, 255201 (2013).

[14] Waser R. & Aono, M. Nanoionics-based resistive switching memories. *Nature Mater.* **6**, 833-840 (2007).

[15] Sawa, A. Resistive switching in transition metal oxides. *Mater. Today* **11**, 28-36 (2008).

[16] Jo, S. H., Kim, K-H. & Lu, W. High-Density Crossbar Arrays Based on a Si Memristive System. *Nano Lett.* **9**, 870-874 (2009).

[17] Martinez-Rincon, J., Di Ventra, M. & Pershin, Y. V. Solid-state memcapacitive system with negative and diverging capacitance. *Phys. Rev. B* **81**, 195430 (2010).

[18] Strukov, D. & Likharev, K. CMOL FPGA: A reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices. *Nanotechnology* **16**, 888-900 (2005).

[19] Linares-Barranco, B. & Serrano-Gotarredona, T. Exploiting memristance in adaptive asynchronous spiking neuromorphic nanotechnology systems. *Nanotechnology 2009, IEEE Nano* 601-604 (2009).

[20] Itoh, M. & Chua, L. O. Memristor cellular automata and memristor discrete-time cellular neural networks. *Int. J. Bif. Chaos* **19**, 3605-3656 (2009).

[21] Pershin, Y. V. & Di Ventra, M. Experimental demonstration of associative memory with memristive neural networks. *Neural Netw.* **23**, 881-886 (2010).

[22] Borghetti, J. *et. al.* Memristive switches enable stateful logic operations via material implication *Nature*, **464**, 873-876 (2010).

[23] Pershin, Y. V. & Di Ventra, M. Solving mazes with memristors: a massively-parallel approach. *Phys. Rev. E* **84**, 046703 (2011).

[24] Linn, E., Rosezin, R., Tappertzhofen, S., Bottger, U. & Waser, R. Beyond von Neumann–logic operations in passive crossbar arrays alongside memory operations. *Nanotechnology* **23** 305205, (2012).

[25] Pershin, Y. V. & Di Ventra, M. Neuromorphic, Digital and Quantum Computation with Memory Circuit Elements. *Proc. IEEE* **100**, 2071-2080 (2012).

[26] Backus, J. Can programming be liberated from the von Neumann style? A functional style and

its algebra of programs. *Comm. Assoc. Comp. Machin.* **21**, 613-641 (1978).

[27] Dawber, M., Rabe, K. M., & Scott, J. F., Physics of thin-film ferroelectric oxides, *Rev. Mod. Phys.*, **77**, 1083-1130 (2005).

[28] Bondurant, D. W. & Gnadinger, F. P., Ferroelectrics for nonvolatile RAMs, *IEEE Spectrum*, **26**, 30-33 (1989).

[29] Razavy, M. *Quantum Theory of Tunneling* (World Scientific Pub Co Inc, 2003).

[30] Simmons, J. G., Generalized Formula for the Electric Tunnel Effect between Similar Electrodes Separated by a Thin Insulating Film. *J. Appl. Phys.* **34**, 1793 (1963).

[31] Traversa, F. L., Bonani, F., Donati Guerrieri, S., A frequency-domain approach to the analysis of stability and bifurcations in nonlinear systems described by differential-algebraic equations, *Int. J. Circ. Theor. Appl.*, **36**, 421-439 (2008).

[32] Cappelluti, F., Traversa, F. L., Bonani, F., Guerrieri, S. D. & Ghione, G., Large-signal stability of symmetric multi-branch power amplifiers exploiting Floquet analysis, *IEEE Trans. Microw. Theory Tech.*, **61**, 1580-1857 (2013).

[33] Traversa, F. L., Pershin, Y. V., & Di Ventra, M., Memory models of adaptive behaviour, *IEEE Trans. Neural Netw. Learn. Syst.*, **24**, 1437 1448, (2013).

[34] Zhirnov V. V., Cavin R. K., Future Microsystems for Information Processing: Limits and Lessons from the Living Systems, *IEEE J. Elec. Dev. Soc.*, **1**, 29 (2013).

[35] Whitehead, A. N. & Russell, B. *Principia Mathematica* (Cambridge University Press, vol. 1, 1910).

[36] A. N. Udipi, N. Muralimanohar, N. Chatterjee, R. Balasubramonian, A. Davis, and N. P. Jouppi. Rethinking DRAM design and organization for energy-constrined multi-cores. In Proc. The Ann. Int'l Symp. Computer Architecture (ISCA), Jun. 2010.

[37] Hennessy, J. L. & Patterson D. *A.Computer Architecture - A Quantitative Approach* (Elsevier Inc, 2012).